

Student ID: \_\_\_\_\_ (no names please)

You may NOT use a calculator. You may use only the provided reference materials. For problems in part II, assume initial memory contents as shown below at the start of EACH instruction. Explain what register/memory location is modified, and give the FINAL HEX value of the modified register/memory location, and the final status of the C, Z flags. Recall that the 'd' bit in a machine word indicating a destination is '0' if the destination is W, is '1' if the destination is the file register. For the 'a' bit, use the assumptions we have used in class ('a' bit is '0' if address in ACCESS RAM, 'a' bit is '1' if not in ACCESS RAM).

Part I: (20 pts)

a. All microprocessors have something that is called a *program counter*. What is this?.

1. Counts the number of bytes in a program.
2. Always holds one of the operands for an ALU operation.
3. Specifies the upper 4-bits of a 12-bit data memory address.
4. Holds the current instruction that is being executed.
5. Specifies the address of the next instruction.
6. Contains the flags that are affected by the current instruction.

b. Give the machine code as a **4-digit hex value** for the instruction:

bsf 0x14F,4

1000 bbba ffff ffff → bbb = 100 because bit 4, a = 1 as 0x14F not in access ram, so  
1000 1001 0100 1111 → 0x894F

c. The machine code 0x31F2 represents instruction? (use 'w' or 'f' for the destination, and 'ACCESS' or BANKED to represent the value of the a bit)

0x31F2 → 0011 0001 1111 0010 , first six bits indicate the rrcf instruction  
rrcf 0011 00da ffff ffff, → rrcf 0xF2, w, BANKED

d. For a 20 MHz clock, how long does it take to execute the following instructions? (give the answer in microseconds)

movff 0x230, 0x110  
clrf 0x002,f

movff is 2 instruction cycles, clrf is 1 instruction cycle. Each instruction cycle is 4 clock cycles, so  $3 * 4 * (1/20 \text{ MHz}) = 12 / (20e6) = 3/5 e-6 = 0.6 \text{ us}$

e. What location in program memory is the first instruction fetched from after a PIC18 reset?

location 0

Location	Contents
0x05E	0xA9
0x05F	0x00
0x060	0x1F
0x061	0xE3

Assume the W register has the value 0xE3 in it, and that initial values of C, Z are both '0'.

Part II. (35 pts) Assume the above memory contents, W register value, initial C,Z values at the START of each instruction.

a. rlcf 0x05E, f

[0x5E] = 0xA9 = 1010 1001 (← left shift)  
 new [0x5e] = 0101 0010 = 0x52,  
 MSB goes into C-flag, so C=1

Circle one: W dest.  Reg. file dest.

New value (hex) 0x52 C\_flag : 1, Z flag: 0

b. btg 0x060, w, 3

7654 3210  
 [0x60] = 0x1F = 0001 1111 (flip bit 3)  
 new W = 0001 0111 = 0x17,

Circle one: W dest.  Reg. file dest.

New value (hex) 0x17 C\_flag : 0, Z flag: 0

c. xorwf 0x061, f

[0x61] = 0xE3 = 1110 0011 XOR  
 W = 0xE3 = 1110 0011  
 new [0x61] = 0000 0000 = 0x00,  
 result is zero, Z=1

Circle one: W dest.  Reg. file dest.

New value (hex) 0x00 C\_flag : 0, Z flag: 1

d. addlw 0x3B

literal = 0x3B  
 W = + 0xE3  
 new W = 0x1E  
 C = 1 because of carry out of MSB

Circle one: W dest.  Reg. file dest.

New value (hex) 0x1E C\_flag : 1, Z flag: 0

e. subwf 0x060, f

[0x60] = 0x1F  
 W = - 0xE3  
 new W = 0x3C  
 C = 0 because of borrow out of MSB

Circle one: W dest.  Reg. file dest.

New value (hex) 0x3C C\_flag : 0, Z flag: 0

All variables in bank0 , no bank switching needed.

(45 pts) PART III. Convert the following C code fragments to PIC18 assembly.

Variable locations are:  $i$  is data location 0x000,  $j$  is data location 0x001,  $k$  is data location 002. Assume the BSR has a 0x0 value in it initially.

If you use a temporary memory location, use temp and assume it is in bank 0. When writing code, you **must use** symbolic names for variable names, register names, and bit names for (i.e. use: `bsf STATUS, C` instead of `bsf 0xFD8, 0x0`). You do not have to show the CBLOCK declaration for variables.

Hint: A common mistake in these problems is to write code that modifies variables to the right of the '=' sign (i.e. for '`a = b - c`;' the code you write somehow modifies  $b$ , or  $c$ , as well as  $a$ ). This is incorrect; make sure that your code only modifies variables to the left of the '=' sign.

Also, recall that '`k++`' is the same as '`k=k+1`'; '`j--`' is the same as '`j = j - 1`', that "`i = j`" is true if  $i$  is equal to  $j$ , that "`i != j`" is true if  $i$  is not equal to  $j$ , "`<<`" is a left shift, "`>>`" is a right shift, '`|`' is bitwise logical OR, '`&`' is a bitwise logic AND, '`^`' is a bitwise logical XOR.

unsigned char i,j,k;

a. (5 pts)  
`k = i & j;`

```
movf i,w      ; w = i
andwf j,w     ; w = j&i
movwf k      ;k = j&i
```

b. (10 pts)  
do {  
 `k = k >> 1;`  
} while (k > j)

```
top
bcf STATUS, C
rrcf k,f      ;k = k >> 1
movf k,w     ;w = k
subwf j,w    ;j-k
bnc top      ; continue if C=0 (borrow,
              ; so k must be > j
```

c. (5 pts)  
k = k + 1 - j;

```
incf k,f      ; k = k+1
movf j,w      ; w = j
subwf k,f     ;k = k+1 - j
```

d. (10 pts)

```
if (j != k) {
    i++; j--;
} else {
    k = k | 0x0F;
    j = 0;
}
```

```
movf k,f
subwf j,w      ;w = j-k
bz     else_body
incf i,f
decf j,f
bra   end_if
else_body
movlw 0x0F
iorwf k,f      ;k = k | 0x0F
clrf j,f      ;j = 0;
end_if
...rest of code...
```

- f. (7 pts) Write an assembly code fragment that copies the contents of locations 0x120 through 0x123 to locations 0x250 through 0x253.

```
movff 0x120, 0x250
movff 0x121, 0x251
movff 0x122, 0x252
movff 0x123, 0x253
```

- g. (8 pts) Write a PIC18 instruction sequence that does  $i = (i \ll 1) + (i \ll 2)$

```
bcf    STATUS,C
rlcf   i,f          i = i << 1
bcf    STATUS,C
rlcf   i,w          w = i << 2
addwf  i,f          i = i <<1 + i<<2
```